AN ARGUS SPECIALIST PUBLICATION

February 1985

A&B COMPUTING

£1.25

FOR USERS OF THE BBC MICRO AND ELECTRON

Mice and Men – A Winning Combination?

Move to the Sound of MIDI – Mixing Music and the Micro

Expand the Electron – Disc Drives are here

PLUS:

A Nightingale sings and travelling beyond the bounds of BBC BASIC – Extension ROM

What it's like to BE Elite!

A&B Computing is constantly on the look-out for well-written articles and programs for publication. If you feel that your efforts meet our standards, please feel free to submit your work to us for consideration for publication.

All submitted material should be printed or typed, double spaced. Any programs submitted should be listed (55 character width emphasised if possible). A cassette of the program alone will not be considered. All programs must come complete with a full explanation of the operation, and where relevant, the structure. We also require the program in machine readable form (cassette, 40 track 5¼", or 3" disc) plus any suitable screen photographs, printer dumps and so on.

All submissions will be acknowledged and the copyright in such works which will pass to Argus Specialist Publications Limited will be paid for at competitive rates. All work for consideration should be sent to the Editor at our Golden Square address.

Volume Two Number

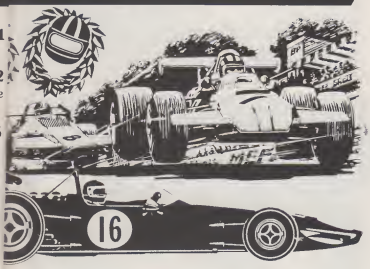
Group Editor: Wendy Palm
Acting Editor: Mark Web
Editorial Assistant: Ione Holme
Advertising Manager: Barry Bingha
Assistant Advertising Manager: Jonatha McGarr
Chief Executive: TJ Conne

A&B

ber Two February 1984

Published by Argus Specialist Publications Ltd., Number One, Golden Square, London W1R 3AB. Tel: 01 437 0626.

All work for consideration should be sent to the Editor of A&B Computing at our Golden Square address.



NEXT MONTH

A computer aptitude test to tingle the nerve ends and tangle your reactions.

Why your micro may show signs of intelligence but not live up to them.

How Mining is catching on as a games activity!

The ABCs compared with immediate business rivals such as the IBM PC and ACT Apricot ranges.

Meet Spiderman looking for the Garden of Eden in a Maze down at Arendarvon Castle — software sector.

The Electron gets a section all to itself.

A&B Computing is published monthly on the first Friday of the month preceding cover date. Distributed by: SM Distribution Ltd, 16-18 Trinity Gardens, London SW9 8DX. Telephone: 01-274 8611. Printed in the UK by Garnett Print, Rotherham and London

Printed in the U.K. by Garnett Print, Rotherham and London.

Phloopy

Jonathan Evans

Is fast tape the answer for your filing system requirements?

If you are fed up with loading files from cassette the chances are either that you are frustrated by its slowness and suspect reliability or that you require serious file handling which is impractical on the tape system (or both).

If you are hesitating to buy a disc drive it may be the price that is putting you off, or the sheer confusion. Do you need standard size or mini-discs, 40 track or 80 track? Do you need a dual disc drive, or perhaps a double sided single drive? And what is going to happen now that the single density chip controller used by the standard Acorn DFS has gone out of production? The observant may have noticed that Acorn's Electron disc upgrade, new ABC computer and Level 2 Econet all involve a new double density chip. Makes you wonder doesn't it?

If price is the problem and you don't already have a disc drive interface fitted to your machine, you might consider a fast tape system as an alternative solution. There are two systems on the market that I am aware of. One is the Ikon Ultradrive (upgraded Hobbit) which I reviewed in the October issue of A & B Computing. The other is the Phloopy, distributed by Phi Mag Systems (Trenoggie Industrial Estate, Falmouth, Cornwall TR11 4RY). There is a considerable price difference, with the Ultradrive retailing at a totally inclusive price of £79.95 and the Phloopy at £147.75 (not withstanding the misleading £99 at the top of their advertisements, which does not include VAT, interface, carriage etc).

The Phloopy's claim for superiority over the Ultradrive is faster file access due to a much faster data transfer rate, on-board microprocessor and 16K (rather than 8K) filing system ROM which avoids the need for a utility tape. However, as we shall see later, the practical file access time on the Phloopy is incredibly variable, according to how it is used. Its main claim over disc is price, though at around £150 it is not much cheaper than the cheapest 100K systems (with interface) that are now being advertised. The only other advantage I can think of is that the tape cartridges are far more robust and easy for handle than floppy discs — a useful point if young children are involved in the use of the computer.

HARDWARE

Anyone who has opened a computer magazine in the last few months will have seen a full colour Phloopy ad explaining its "byte wide" head, ie eight track recording, which has enabled them to achieve the astonishing data transfer rate of 10K per second, which is comparable to disc.

Unlike the Ultradrive (which is reel to reel), the Phloopy uses a continuous tape loop, but one which they assure me is superior in design to the much criticised Sinclair microdrive. Apparently the latter uses a Centre Tap spindle, whereas the Phloopy uses a Bin Loop, for the benefit of any readers who might appreciate the distinction.

The tape actually has extra tracks which carry redundant information to allow automatic error detection and correction by the built in microprocessor. It uses random rather than serial access, which means that a complete map of the tape is read each time a filing system command is issued, and a file can be saved and retrieved in separate chunks distributed throughout the tape. For this reason, the tape never needs to be compacted, since all the free space from the front of the tape is automatically filled.

A nice feature of the Phloopy is the on-board microprocessor which performs some operations, such as formatting a tape, while returning control of the BBC computer to the user. Although the system needs to return the recording head to the splice following each filing operation, this is again handled by the microprocessor without occupying the computer's time once the save or load has been completed.

The Phloopy connects to the disc port and requires the fitting of an interface. I didn't have to do this myself since I was loaned a suitably equipped micro for the review, so I can only pass on a summary of the very clear and detailed instructions in the manual (the documentation is generally very good).

Two 14 pin connectors are inserted in the IC79 and IC80 sockets and a much larger controller board into the IC78 socket — the text suggests that lining up the pins might be a bit tricky here. There is, of course, a filing system ROM to be fitted into one of the sideways ROM sockets, and a bit that will frighten some readers — a need to cut the wires leading to resistors R22 and R23. The final step is to connect the ribbon cable to the disc port and the power supply cable to the socket provided on the BBC machine. Although the instructions are clear, with diagrams, some people will inevitably lack the confidence to do it themselves (or be worried about warranty), in which case a dealer's fee must be added to the cost of the system.

OPERATION

The Phloopy loads and saves files in a completely automatic fashion and responds to all the usual filing system commands (*CAT, *SPOOL etc) in addition to permitting use of the BASIC commands for random access such as PTR# and EXT#. A full list of Phloopy's additional commands (all in ROM) is given in Table 1. The manual gives a detailed description of all filing system commands, both general to the micro and specific to the Phloopy. A disc user (Acorn DFS) looking at Table 1 will spot the addition of the useful *VERIFY and note, in particular, the absence of *COMPACT and *BACKUP. The former, as I have explained, is not needed but the latter would have been useful especially because *COPY can only handle one file at a time with no wildcards etc.

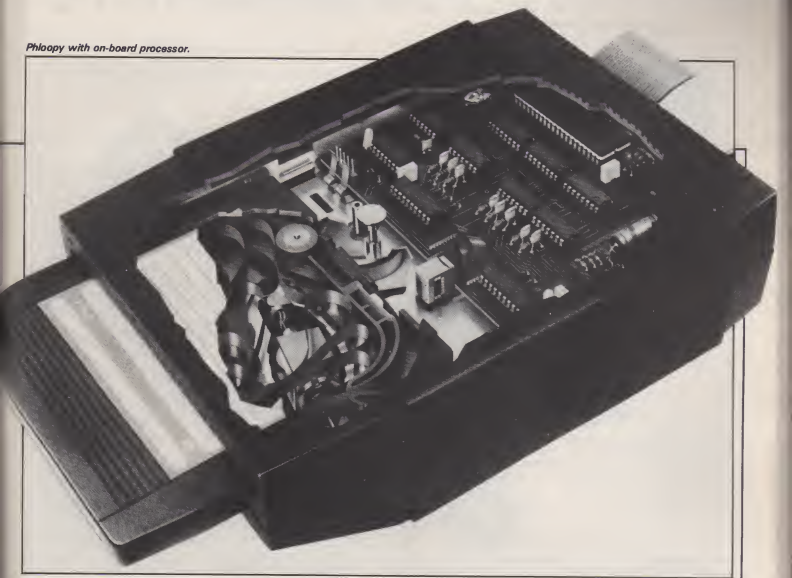
We now consider the critical question of the speed of the system which, at the price asked, one would expect to improve considerably on the Ultradrive and approach that of a disc. First impressions are that this is indeed so, until one investigates a little more carefully.

A standard 100K Phloopy cartridge takes 13 seconds to go around the loop, so this is the maximum time that it can take to load a single program. By comparison the Ultradrive can take up to 45 seconds when formatted with two catalogues on each side of the (120K+) tape, and it can take up to 45 seconds to change catalogues if you start at the wrong one. Hence, the Phloopy shows a big advantage over the Ultradrive but some inferiority to disc unless the program loaded is the first file recorded on the tape.

The big speed advantage of the Phloopy over the Ultradrive may, however, be lost or reversed if a program loads in several parts. This is because the Phloopy can only pick up one file on one spin of the loop, and cannot chain a series of programs recorded in correct sequence at one pass. This does, at least, have the advantage that one does not need to worry about the order in which programs are saved. However, we can begin to see why there is much more involved in file access time than data transfer rate.

We now come to the major weakness of the Phloopy, which is the way in which it handles data files. All filing systems load and save data files slower on the BBC micro than LOAD, SAVE, *LOAD or *SAVE operations, because the filing system is switched off between blocks of 256 bytes. The effect on the Phloopy is, however, much more dramatic than with other systems.

With a single data file open, as when using Wordwise, it



Phloopy with on-boerd processor.

manages to pick up (or lay down) about 1.5K and then must do a complete loop before it can do any more. To test the effects of this I loaded the identical smallish (6.3K) Wordwise file on several different filing systems. A Teac 80 track disc drive took a surprisingly long 14 seconds, while domestic cassette loaded the file in 140 seconds.

The Phloopy took 70 seconds and lost comfortably to the Ultradrive which loaded in 50 seconds (all figures approximate!).

Phi Mag have responded to the Wordwise problem in two ways. One is that they have produced shorter loops for development work. The same file loaded in 32 seconds and 20 seconds on a 50K and 25K loop respectively. The snag is that these short tapes come very expensive for the amount of data they hold. The standard 100K cartridges are fairly pricey in my opinion — £18.11 for 5 or £4.25 bought singly. Both 25K and 50K tapes will sell at £16.95 for 5 or £4.03 individually. The company also tells me that they are developing their own wordprocessor for the Phloopy which they plan to supply free of charge on cartridge when the whole system is bought. I have, of course, no idea how good this program will be, but I understand that it will overcome the slow save/load problem. At my suggestion, the company are now testing View for its suitability of use with the Phloopy.

Wordwise is not the only user of data files of course, which may be used for saves in games programs and also in users' own BASIC programs for data logging etc. This is where I have to give the really bad news. The Phloopy, like disc and Ultradrive, claims to be able to handle up to five open files at once. Unfortunately, once a second data file is open, the system can only handle one 256 byte buffer per spin of the loop on any open data file.

In one test I loaded a small 3K data file from a BASIC program first with one file open and then with a second opened. The Ultradrive took 28 seconds in either case. The Phloopy (100K tape) took 32 seconds with one file open, but an horrendous 2 minutes 41 seconds with two files open. I am forced to the conclusion that for all practical purposes the Phloopy is a non-starter as a multiple file handler.

SOFTWARE

Once you upgrade your filing system you will run into problems with commercial software. Some programs are available on disc, but many, especially games, are not. Despite the claims of the advertisements it appears that no commercial software houses have as yet committed themselves to supplying software on Phloopy cartridges. Transferring tape programs to disc runs into two problems, program protection (if present) and use of memory, since the disc system sets page up from &E00 to &1900.

Similar problems arise with the Phloopy since it normally sets PAGE to &1600. The problem is a little easier than with disc since a program can be loaded direct to &E00 if it does not then attempt to load or save another file, and to &F00 allowing chaining of other programs etc. However, any program which loads below &E00 will need to be "downloaded", ie loaded in higher in memory and then shifted down in software before running.

Phi Mag have provided supplementary documentation to their customers on how to download machine code, as well as BASIC programs. They have also provided a command in their ROM, *TRANSFER, which

CONTINUED OVER

transfers single files from cassette to Phloopy automatically. Whilst this will transfer a locked file, if protection is found on the original it is also placed on the Phloopy copy. Whilst the company presumably did this to avoid accusations of aiding piracy, it was actually a mistake, in my view.

They seem to have entirely overlooked the fact that most machine code games load below &E00 and thus cannot run on Phloopy without downloading. Since a protected Phloopy file can only be *RUN, their (presumably) frustrated customers have no means of *LOADing the copied file in order to download it. There is also a bug in the *TRANSFER routine, which sometimes puts protection on to harmless BASIC programs which then refuse to CHAIN. Such programs can of course be transferred by *TAPE, LOAD, *PHLOOPY, SAVE etc, but the utility provided is more convenient and should work as documented.

CONCLUSIONS

How good a buy is the Phloopy? The sensible comparisons seem to be with the Ultradrive, which is much cheaper, and a disc drive which is significantly more expensive (at least one of a quality worth buying). Firstly, Phi Mag are to be congratulated on achieving a superb data transfer rate equalling or bettering disc, and on the use of an on-board microprocessor which has many advantages. They have convinced me, in principle, that fast tape systems could provide some genuine competition in performance with floppy disc systems.

Unfortunately, their current software design does not seem to me to permit the hardware to realise its potential. When programs chain in sections or when one data file is open (as in Wordwise) the system's performance is way below that of discs and, if anything, inferior to the Ultradrive, unless one is prepared to work with the very short and expensive tape loops. Worse still, in multiple file handling operations in which the disc system excels and the Ultradrive is workable, the Phloopy is effectively useless.

How many people need multiple-file handling, the manufacturers will retort. Well, it is needed in serious database manipulation, for example, to keep index files to speed up the computation of access into large masterfiles. True, commercial software written to use such facilities is aimed at disc users, but what of programmers who wish to do such operations themselves? I would imagine that secondary school usage in both teaching computer science and laboratory applications will require this facility, so the Phloopy must concede this potentially lucrative market entirely to discs. The Phloopy might appeal to primary schools because of its relatively fast program access time and robust cartridges. Whilst much educational software makes use of the datafiles, these are generally single file operations because the software is written to work with ordinary cassettes.

What of home owners? If they are primarily Wordwise users or programmers wishing to do extensive data file handling, I would have to recommend that they upgrade to discs if they can afford it, or to the Ultradrive if they cannot. If their prime interest is in writing programs which do not make much use of data files then the Phloopy will provide very good performance. Whether the system is a good buy for games players is debatable: (a) it's a lot of money to pay for the privilege of loading your games more quickly, (b) there is no guarantee as yet that commercial games will be sold on Phloopy cartridges, (c) a fair proportion of cassette based programs will prove difficult to transfer.

One thing is for sure, no one should buy the Phloopy without knowing what kind of use they want to make out of the micro and checking carefully whether the system will be suitable. I think this is unfortunate since one of the big advantages of the BBC micro is its flexibility. Thus owners might initially buy one for games or educational software but later get into wordprocessing or serious database management. If they have bought a relatively expensive filing system they are entitled to expect it to cope with the range of filing activities provided by the machine without drastic loss of efficiency.

Table 1

*** Commands in Phloopy ROM**

* COPY	Copies a single file to another tape either on the same drive or a different drive
* DELETE	Deletes specified file
* DRIVE	Selects drive
* FORMAT	Formats a blank tape
* HELP PHLOOPY	Lists commands
* INFO	Provides parameters of specified file (load address, length etc)
* LOCK	Locks files to prevent accidental overwriting or deletion (equivalent to * ACCESS on disc system)
* PHLOOPY	Selects Phloopy filing system
* RENAME	Renames specified file
* TITLE	Gives a title to a tape (optional)
* TRANSFER	Transfers a file from domestic cassette recorder
* UNLOCK	Reverses * LOCK
* VERIFY	Verifies saved file with contents of memory

Bookshelf

Disk Systems for the BBC Micro by Ian Sinclair. Published by Granada. Price £6.95.

This book on utilising disc drives alongside the BBC has been around for a few months but is now particularly relevant as more and more Beeb owners turn to discs as a storage medium. Ian Sinclair is steeped in detailed knowledge of how disc systems work with most micros that support them and is an ideal candidate to produce an introduction for the BBC.

Sinclair's knowledge of other micros (Tandy, Apple) enables him to go as far as advising on transfer of text (ASCII) files from one computer system to another (the Beeb's) — a complex process. At the other end of the extreme, great care is taken to explain in simple terms how the software handles disc storage and how the user can handle the software.

A great many disc drives are being purchased independently of the DFS, and documentation in most cases is poor. There are also two viable alternative filing systems by AMCOM and Watford Electronics. The author continually cites examples from these systems, usually in more detail than for the Acorn version. A case in point is the chapter on using disc utilities. AMCOM's package allows for direct disc surgery, with sector editing and the like, and is an ideal example. Much of the advice applies to using Disc Doctor or Disc Recovery.

An introductory chapter about discs and drives leads into a look at filing system commands. There are lots of examples to try out as you go along and hints on the thorny problem of tape to disc transfer. "Digging Deeper" investigates some of the more subtle uses of wildcards, COPY, BACKUP, RENAME and so on. It also explains some of the problems you might encounter, like "can't extend" messages — when a new, longer version of a file is SAVEd with the same name, but the disc has not reserved extra space.

The book covers machine code programs, saving chunks of memory, *BUILDING !BOOT files, text files (a whole chapter mainly concerned with Wordwise) and filing techniques. The chapter on the latter is clear, comprehensive and full of examples; the best contribution to a very useful guide for disc users new and old.

Interfacing the BBC Microcomputer by Colin Opie. Published McGraw Hill. Price £8.95.

The dazzling silver cover of this book disguises a highly technical but clearly presented look at interfacing the BBC. The book makes full use of line drawings, illustrating both general architecture and complex circuit and program flow.

There are helpful tables (especially memory maps) in the text and the appendices. These appendices also contain the circuits and board layouts for the hardware projects introduced during the last section. Where relevant, chapters are concluded with a useful bibliography.

First off, there is an introduction to the basic micro system and a look at how the various elements communicate. It looks at 6502 operation, serial/parallel communication, Boonlean logic, interrupts, A/D conversion types, TTL devices and buffers. Pretty comprehensive as you can imagine. Anyone coming to interfacing anew need not worry since the author takes care to explain the basics before going on.

Time for FRED, JIM and SHEILA, those important I/O related pages of BBC memory. Colin Opie rightly emphasises the need to use O.S. calls rather than directly changing memory. The second processors are with us and changing software can be hard work.

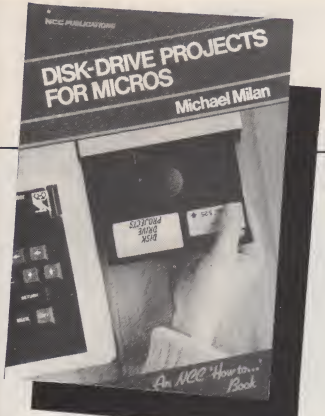
We get excellent and lucid accounts of the user and parallel printer ports, with circuit diagrams and pin functions. Same treatment for the analogue to digital port. For this part of the book you will need a general understanding of symbolic representation of components in circuit diagrams. The RS423 and 1Mhz bus are now investigated. The usefulness of the 1Mhz bus for control applications becomes clear and its design is explained in detail.

The hardware now over, Part 2 starts on programming it. There is clear tabular information relating addresses in memory to registers in the hardware. Each I/O port is treated individually and read/write routines listed in BASIC and Assembler forms. There is also line by line explanation in some cases.

The reading of this section results in a comprehensive study of programming the devices which link the Beeb to the outside world, analogue or digital, serial or parallel.

For Part 3, the publishers have got together with Watford Electronics to supply a motherboard and application boards so that theory can be put into practice. Real results do wonders for the concentration. The ready availability of hardware also makes the book a useful guide for self study or for work in the classroom.

If you are thinking in terms of using your BBC for control applications or you would just like to understand more about what is going on when you plug in your interfacing box, or even when Acorn come up with their home control system, then you can't really do better than getting hold of this book on the subject.



Disc Drives Project for Micros by Michael Milan. Published by National Computing Centre. Price £5.95.

Don't be put off by the title. No mention of a BBC? Nothing about the Plus 3? Whatever it says on the jacket, the picture is a dead giveaway. The whole book is based upon handling disc based information with a BBC Micro and Acorn DFS. If I owned another computer and bought this book on the understanding that there might be some worth in it for me, I might be very disappointed. As a BBC owner with disc drives, I can feel delighted.

The book covers everything you need to know about actually putting your disc system to work, it does not go into all the hardware and software details but helps you get your system up and running and only then pursues some more advanced features that discs provide.

There are general points about the sort of hardware available, and pointers on what you might need. The software side starts simply and extends up to the heights of software subtlety, random access filing.

This book would in fact make ideal documentation for any one of the current multitude of disc drives available for the BBC and Electron Plus 3.

This is one of a series of NCC publications which thankfully makes a point of not jumping in at the deep end, leaving the reader, and proud owner of new disc drives, drowning in *commands, hex locations, sectors and cryptic abbreviations.

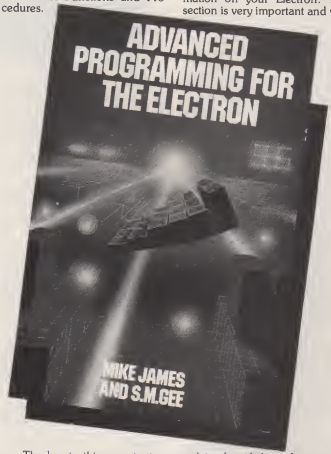
Each new element of disc use, from simple saving and loading, to complex file handling, is approached as from new. Throughout the book examples of what you will see on the screen are provided in the form of screen dumps. There are also a number of example listings which can be attempted by the reader as he/she works through the various commands now at his/her disposal. Having the power of disc drives available means that the user is tempted to do many things which seemed tedious when working with tape, like saving and loading screens.

High resolution screens take up quite a bit of memory and loading them in block by block from tape is boring. With discs it takes just a few seconds and is easy to execute. The book takes us into the areas of *SAVE, *EXEC and *SPOOL. Discs are more fun. Next up are serial files, another area where the speed of drives makes all the difference. This section and the next, on Direct Access Files, are full of examples and conclude with complete listings of useful filing programs.

The final project is a graphics drawing suite, which uses discs to store the current screen image. Two quick appendices take in error messages and "other computers". Don't doubt it, this is a BBC book and a very good introduction to disc handling on all levels.

Advanced Programming for the Electron. by Mike James and S.M.Gee. Published by Granada. Price £6.95.

This book sets out to tackle the problems of writing larger BASIC and Assembler programs on the Electron. The authors' approach is designed to produce organised programs, carefully planned and designed before coding. They look for reliability through the use of the natural structures of BASIC, like REPEAT...UNTIL and FOR..NEXT, and through the use of Functions and Procedures.

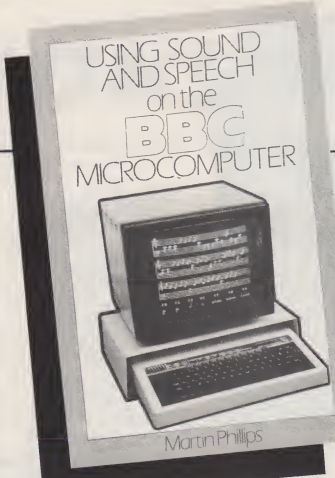


The key to this organisation is the modular structuring and building of programs — a theme which crops up throughout the book. The authors go into "stepwise refinement", testing and perfecting the individual components of the program. They discuss the use of parameters and local variables as well as the physical appearance of listings, formatting and use of REMs.

On the Assembler side, there are discussions on the natural structures of assembly language, details of how the Electron assembler can be used and a few paragraphs on Macros and some of the problems associated with implementing them.

As the book gets more involved, and the O level Maths get a bit strained, we move onto data types, arrays, look-up tables, handling a stack structure; everything you need to know about storing and accessing information on your Electron. This section is very important and well explained, with lots of examples and step by step runs through.

CONTINUED OVER



These techniques are used by every program worth its salt, from databases to arcade thrillers. A section follows on both BASIC and Assembler control of sequential filing.

An interesting contribution appears in the form of "Making Programs Work" — pretty important! It looks at the testing of programs, the location and eradication of bugs, common errors and program presentation.

Chapters eight, nine and eleven are based upon full-scale programs, Spelling Checker, Execution Tracer and Dissassembler. Each is built in stages, including design. All have major elements of Assembler (spelling checks are useless if they are slow) but use BASIC to hold them together and present them in an organised and understandable fashion.

Unfortunately the Spelling Checker, though it works, is, by the authors' own admission, not a completely finished version. Still, if you take all the rest of the information in, you may well be in a position to do something about it! The Execution Tracer takes up the theme of debugging and could be a useful program for the new programmer.

On either side of the Disassembler project are chapters on the MOS and on fundamental background in Bits, Binary and Boolean. The Machine Operating System and the part it plays in the everyday functioning of the Electron is explained. Work on OS calls and interrupt handling lead to the construction of a background clock.

The final chapter looks at binary hex and logic in relation to the way Acorn BBC BASIC and Assembler handle numbers.

The Disassembler itself does not have enough facilities to be really useful (but is open to improvement), but introduces some more complex programming techniques and the final result is good enough for "Leafing through" Electron software.

This book is for the programming enthusiast. Although the techniques discussed, and practiced throughout, would enable the writing of efficient programs of all types, the approach is entirely from the point of view of writing code. It will be beyond anyone who is not already an interested and fairly experienced programmer. For the ambitious Electron programmer, who wants to acquire the skills of a professional, this book is an excellent start.

Using Sound and Speech on the BBC Microcomputer by Martin Phillips. Published by MacMillan. Price £6.95.

Perhaps it is the complexity of the SOUND and ENVELOPE statements in BBC BASIC or perhaps the romantic idea of sitting back for an evening's musical entertainment with your BBC, but sound/music books seem to spring up at regular intervals.

This one by Martin Phillips covers much of the usual ground, expanding upon the official explanations of SOUND and ENVELOPE, looking at the physics of sound and the programming of music. It also supplies the first (in book form) independent look at the Acorn speech synthesis add-on.

The explanations of sound are an interesting path into the BBC Micro's own production of pitch, amplitude and timbre. The author has written some nice programs to graphically demonstrate the production of wave forms, including a representation of air displacement. This sort of technical information is interlaced with three and four line program examples as well as larger listings. As explained in the introduction, considerable care has been taken to provide well-documented and readable listings. The technical appreciation continues with a fascinating investigation into the whereabouts and functioning of the sound buffers and an highly accurate table of measured frequencies for note production, highlighting the inadequacies of the sound chip.

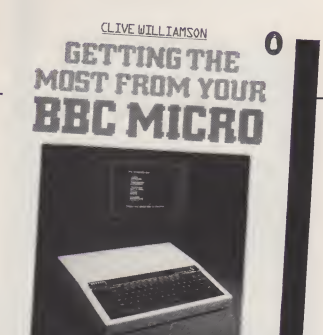
There is a brief appendix on the mixer and amplifier circuits and reproduction of the circuit drawings for sound and speech.

A chapter on assembly language control of speech is disappointing although it hints at the possibilities. Ian Birmbaum's assembly language book in the same Macmillan series will prove more fruitful reading for those venturing into this area, alongside the user guide. The use of macros for generating both speech and sound (something I find a natural use with tables of encoded notes or parts of speech) is not covered. The author touches on the use of event handling in conjunction with speech, resulting in a "talking typewriter" program.

Music making on the BBC is difficult and the chapter in this book does not really investigate the theory or practise of programming music. Music and musical notation are introduced for the layman and much of the programming goes into producing graphical representation of (outmoded?) notation. There is a full scale program to play tunes and display the appropriate notes on a stave at the same time, a useful educational exercise.

The chapter on speech synthesis uses much the same approach as with sound. We get an introduction to the hardware, the methods of reproducing human speech, an analysis of speech itself and how the problems of storing it in computer memory have (to an extent) been overcome. There is a detailed look at how to use the words and parts of speech provided by the Acorn system, culminating in an example of how to implement a full speech application into your own programs. Speech is amazingly popular with children especially and does much to enhance educational programs. Games scores and high-score tables can similarly benefit. Reading this chapter will probably persuade a good many people that speech synthesis is a fair investment if you are willing to do some programming.

Informative on some points and original on others, this book would make an ideal first book about using sound and speech on the BBC. If you have already gone into sound, or already own the speech synthesis system, then much of the information here is already available to you. Nor is this book for music makers unless to gain a more technical understanding of the internal operations of the sound chip and operating system software. If you have always fancied delving into sound or speech then Martin Phillips' book will prove a precise and informative partner.



Getting the Most from your BBC Micro by Clive Williamson. Published by Penguin. Price: £5.95.

Clive Williamson has done a fair job producing yet another follow up to the BBC User Guide. New BBCs are continually being purchased and new owners are still apparently being bewildered by Acorn's own manual.

If the user were to actually "take on" the User Guide then this sort of book wouldn't have much of a market. Most of the information here is paralleled in the guide. Yes, there are more examples and the flow of the book is more logical than the guide, but there is not enough difference to make it in any way an addition to the information or its availability.

There is a gap in the standard documentation which accompanies the BBC — filled nicely in the Electron package by Yazdani's Start Programming — but this new attempt to fill it is a lot less interesting especially on sound and graphics than its already established rivals form Prentice-Hall, Granada and Addison-Wesley.

Adventure Games for the Electron by A.J. Bradbury. Published by Granada. Price: £6.95.

This very impressive book is a bit more technical than its sci-fi cover suggests but this should not deter the Electron (and, I see no reason why not, unless Granada are going to come up with a BBC version, the BBC) adventurer from going straight out and buying it. If you bought Peter Kilworth's Penguin/Acorn guide then don't worry. There is some overlap in subject matter, naturally, but with both you will be as clued up on the subject as most.

There's the obligatory history lesson, Standford Research versus Massachusetts Institute of Technology. Collossal Caves versus Dungeon. The latter's Infocom language is compared with Melbourne House's Inglish, found in the Hobbit. Language recognition is all part of the believability of Adventures.

Later in the book, chapter eight goes into the tokenising and crunching techniques needed to squeeze "language" into a micro Adventure. This chapter presents some very interesting ideas and demonstrates them with a set of programs for encoding and decoding text/descriptions — similar methods, we are told, to those used by Level 9.

For those new to programming, there is a line by line analysis of each largish program. The method used is one of assigning character groups to ASCII codes above 127. A string analysis program lists the most common letter groupings and this information is used by the encoder.

Strings are stored above HIMEM in byte arrays and accessed with a pointer table. Electrons with only Mode 6 and not a lot of memory to play with, will especially benefit from this sort of crunching.

However, Adventure Games is not just about the coding. The author deals with the creative side, the necessary elements to maintain interest, the plot, problems and clues. He also looks at planning, internal consistency and comprehensibility. Creating characters, keeping track of movement, not just of the player but other creatures, saving games, analysing player input; all these aspects and more are discussed with programming examples. Finally there is a chapter on limited use of sound and graphics.

It's fairly obvious that the author did not develop all his ideas on an Electron, or even a BBC. As an ardent adventurer, steeped in the American game, he reveals himself as an Apple owner. No matter. Even if he does use subroutines rather than procedures. Electron adventurers would not want to miss out on a wonderfully enthusiastic, breezily written and technically comprehensive manual on Adventure writing.

Drawing your own BBC programs by Jonathan Grieg. Published by Century Communications. Price £6.95.

Despite the rather strange title this is a good, interesting book on the subject of BBC and Electron graphics facilities. The 160 pages of main text split up into six main sections, each covering a major point of the machine's capabilities or of the mathematics involved in using the graphics to get the effect that you need.

The first section "High resolution graphics" explains how the various PLOT and DRAW options work and how data can be scaled, translated and rotated to change its effect on screeen. This section involved some mathematics and despite my being decidely rusty on these matters it soon started to make sense and fall into place.

The next section is about "Block graphics" which are usually referred to as character graphics in the case of the BBC and Electron. Once again there was a good introduction on the subject of character definition and movement around the screen. Each part of the discussion is punctuated with short procedures or program lines ready for you to try the effect and experiment as you go along.

Then comes the section on colour and animation which starts appropriately enough with an explanation on how colour screens work. There follows an explanation of the various colour options available and their effects on the screen, including the use of truth tables for logical operations. This chapter closes with a discussion of the various types of animation that can be achieved by colour manipulation, a common technique on the BBC machine.

The following sections deal with the uses of graphics in other programs such as graphs and charts and covers the various methods of circle drawing that you might like to try. The section on three dimensional graphics was rather too mathematical for my liking but there were plenty of examples to try and a number of well defined formulae to use when writing your own programs.

The book closes with a chapter on the block graphics of Mode 7, therefore only of interest to BBC owners. This section was well written and contained a great deal of interesting information well presented with little programs to try.

All told this book provides a good introduction to the mathematics and programming

CONTINUED OVER

of graphics on these machines and whilst it isn't at a high enough level for those to whom the mathematics is relatively easy, I can see a number of young people and indeed their parents getting a good grounding in the subject from this text. Well worth the £6.95 for a good, steady easy-to-read book.

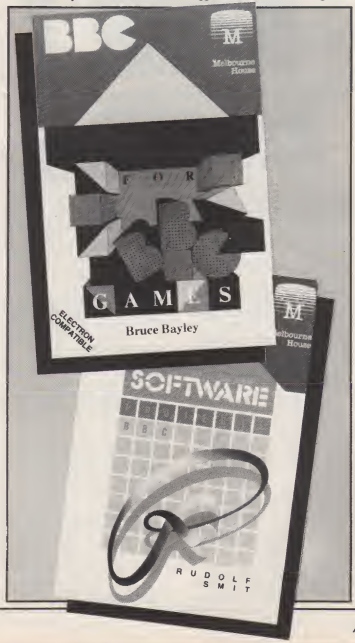
BBC Software Projects by Rudolf Smit. Published by Melbourne House. Price: £6.95.

It strikes me, after reading this book, that you shouldn't assume that because a company has a good reputation for producing software of distinction that they can also produce books about programming. This is a poor book by any standard, for all the reasons that I give below, but to have such a book published under this company's name is a double tragedy.

This is a teaching text with the avowed aim of giving you a good grounding in programming and in project management. It takes, what I am presuming is, a rather unique approach to its teaching. It proposes a project and then gives a full description of the program required but doesn't actually provide the program. That is where you come in and the book is therefore comprised of six projects all in a half finished state. This is a good idea and if done well would be an efficient and involving way of learning to program. Unfortunately here it is not well done and the book is riddled with contradictions and mistakes, enough to leave me feeling that it was rushed onto the presses without anywhere near enough checking and attention to detail.

There is a program that makes no sense at all because part of it is missing, there are wrong line numbers mentioned in the text and there are even mistakes in variable names. All of these could be simple typing errors but it doesn't stop there, there are errors of fact included too. The book states, quite firmly, that you cannot have DEF FN code, ie used defined functions, that aren't in the first procedure called. This is nonsense, if they are initialised in this way they work faster, but to say that they cannot be anywhere else is fanciful. Add to this the fact that the book is about good structured programming yet there is not one mention of input validation and you have real problems.

The projects in the book claim to be interesting but, to be honest, they are boring and I think that most readers will find them this way too. The first three projects are simply variations on a theme, data storage, and simply change the method from project to project. The level of the text is low as you might expect but there are a number of restrictions placed on the programs that you have to write. A good example of this is the way that variable names are specified, in order that the rest of the program can use the same names. This too would be fine, if the author actually practised what he preached — he suggests the use of long and meaningful variable names yet regularly uses single letter names himself. That isn't the end of the list but I've run out of space. This book is based on a good idea, in teaching terms, but it has been so badly thought out as to be rendered useless.

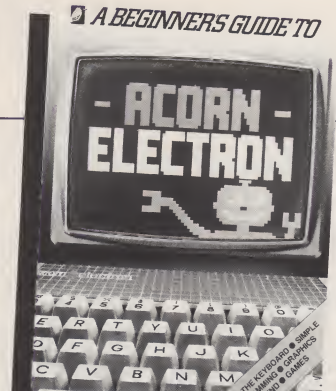


Building Blocks for BBC Games by Bruce Bayley. Published by Melbourne House. Price: £8.95.

This book, "for all BBC enthusiasts" according to the blurb, suggests that one way to make games writing easier is to have a set of pre-programmed modules to do the common jobs within a program. Then it goes on to show how the modules can be used to provide "exciting nd original games". Far from being exciting and original, games like River Frog, Luna Landa, Simon and Brick Out are more likely to elicit a groan than a smile in this day and age.

The first section of the book on programming and how to manage a game writing project is fairly good and accurate. The diagrams are clear and the use of good programming structure is emphasised. I do wish though that authors wouldn't confuse "Top — Down design" with "Structured Programming", there is a difference. The last part of this chapter contains a procedure called "chexsum" which is a method of checking programs as they are input and seeing where any typing mistakes might be. This is a great idea for machines that have a poor error reporting system, which cannot be said of the BBC or Electron machines. With these machines the procedure is almost redundant as the reports tell you a great deal more than a checksum ever could. It helps the writer however, each program is followed by the full "chexsum" list and by my quick calculation that takes up over 20 pages of the book!

The next section contains three utility programs, a character definer, a sound definer and a background builder. The first two are both reasonable but are far less well organised and capable



than a number of such programs that you have read in magazines. The character definer will give you the VDU 23 lines for the characters but will not allow any colour changing within a character. The sound aid allows you to redefine envelopes but only the pitch envelope section. This is probably because the amplitude envelope is fixed on the Electron but means that it is less flexible than most others I have seen. The background aid is a very strange invention for developing DATA lines to be used to draw backgrounds in games.

The actual program modules themselves are a surprising collection to say the least. They comprise modules to move a character around the screen, clear a message off the screen, build up a deck of cards, pick a card from the deck and build backgrounds from DATA lines. Apart from a machine code version of the move module that is it, five modules. No mention of high score tables, multicoloured sprites, title screens, instruction routines etc. The routines seem to work alright but they leave a great deal to be done by the programmer. In a matter of minutes I could produce a list of modules of far more value than these.

Even the author doesn't exactly use them extensively. Of the seventeen finished games in the book, five don't use any of the "building blocks" and only a couple use three of them at once. By far the most valuable is the movement procedure, which is called different names in some parts of the text! This is rather too little a core on which to base a book and as such I could only recommend it to those who are interested in having a collection of simple listings to type into their machine. Don't think that you will learn a great deal from doing so however, there is a very minimum of program documentation. Save your money.

A Beginner's Guide to the Acorn Electron by Richard and David Graves. Published by Kingfisher Books. Price: £2.50.

This simple and practical guide provides a useful introduction to learning to use the Electron. It was written by a teacher with the help of his 11-year-old son because they found many of the standard books on introducing programming were far too complicated. It gives step-by-step instruction on writing simple programs, using number and string variables and producing sound, colour and simple graphics.

The book is written in a light style and its colourful pages are filled with humorous illustrations. While it will be helpful for beginners of any age it is clearly aimed at young people. The reader is taken through the chapters in stages from initial familiarisation with the machine to a final summary of programming words, and a series of problems and solutions runs through the book. Each chapter ends with "A reminder" to summarise what has been taught.

Although the book gives clear explanations of what it is teaching, the text is as brief as possible and filled with practical examples to keep children interested. As early as the second chapter, three very short but complete programs are given to show what the computer can do with colour and sound. The theory is detailed in later chapters but hopefully readers will be encouraged to continue through the book if they know what they have to look forward to. Longer programs in the later chapters include a guessing game demonstrating both number and string variables, a birdsong program using procedures and a "ghosthunt" program which contains examples of everything taught in the book.

At just £2.50 this book would make an ideal gift for any young person who is beginning to realise that there is more to computers than just playing games. If they can understand and appreciate all that is explained, they will be well on the way to writing their own simple programs.

The Better Guide To The Acorn Electron by Geoff Bains. Published by Zomba Books. Price: £3.95.

This book, described as the "alternative user's manual", explains how to use the Electron without assuming any previous knowledge but only an interest in the subject. It is written in a light, straightforward and conversational style and takes the reader through the most popular aspects of programming, with plenty of encouragement to experiment further.

The guide covers every topic in detail, starting with a simple explanation of how a microcomputer works and continuing through using the keyboard; words and numbers including real, integer and string variables and arrays; using the screen for display; modes and colours; sound facilities, music and noise; graphics; data-day programs, animated graphics and finally converting BBC programs to run on the Electron. The book is concluded with two appendices explaining how to load, save and catalogue programs and how to compact programs to save memory and speed up running time.

Each chapter starts with an illustration but after that the pages are crammed with text, broken up only by programs and occasional diagrams and tables. The book's design will not appeal to young people but those who want to read as much as possible will appreciate the value for money it offers. Any beginner with enough concentration and interest in the book could go through it page by page and feel they had learned a great deal at the end.

It is not a very easy book to pick up and put down, mainly due to the poor presentation. There is no contents page or index and no summary of what each chapter explains. Even the chapter headings do not shed much light on their subject and there are no sub-headings to break them up. It becomes very annoying when you can vaguely remember reading something useful and want to check it again but have no idea where to find it, and chapters titled "Graphic Details" and "More Graphic Details" don't really help. This is a pity because the text is enjoyable to read and makes the subject matter interesting and appealing.

The book will be useful to anyone who has bought an Electron and wants to learn how to use it fully, but finds the User Guide rather dry. While it could be improved simply with better presentation, it is well written and

CONTINUED OVER

provides very clear and helpful instruction on the various aspects of simple programming.

Basic on the BBC Micro
Further Basic on the BBC Micro
Assembler on the BBC Micro, a Beginner's Guide
by Richard d'Silva. Published by M.U.S.E. Price £1 each, £7.50 for a set of ten, or bulk rates for 30+.

Micro Users in Secondary Education is a well-respected body, leading as it has the development of micros in schools. These pamphlets are produced as a very inexpensive source of teaching material for a group of children beginning to learn on the BBC Micro. The idea, obviously, is that each student should have an individual copy of the booklet, and in that way it turns out to be a very cost-effective scheme.

Mr d'Silva is a teacher, and obviously these booklets reflect the way in which his teaching has ironed out all the wrinkles that most teachers find in their working. My one source of a little concern is that some of the Model A, but there again, I suppose there must be a few of those still around. Each TASK appears to be about right for a lesson, albeit a slightly short one, and appears to be quite self contained. This is followed by a section of further ideas which the student should try. As with all activities of this nature, it is during the experimental stage that real learning takes place. Short instructions are given, but these should generally be sufficient. There will presumably be a teacher around anyway to deal with problems. Again more information on the commands can be found from the User Guide.

I like the level of advancement between subsequent lessons in the first booklet. This continues into the second Basic book, but I think many students might find the TASKS in the Assembler booklet a little demanding. All in all, this is a novel idea, and many less confident teachers might well benefit from Mr d'Silva's buoyant style. Even those teachers with their own strengths might find his ideas a welcome alternative, being clear yet concise. Above all, I am sure the student would find the process enjoyable and would benefit from having the text constantly available.

Business Applications on the BBC Micro by Susan Curran and Margaret Norman. Published by Granada. Price: £7.95.

I cannot recommend this book to anyone who is seriously thinking of using their BBC for business and I think it would bore the rest of the Beeb user community stiff.

It is almost entirely a descriptive work. It describes the general functions of wordprocessing, keeping accounts and so on. This is partly culled from a similar book for the Commodore 64. It also describes various pieces of commercially available software for the BBC plus DFS and the BBC plus Torch Z80 disc pack. There is software from Gemini, MicroAid, BBC Publications, Acornsoft and Software for All as well as many of the CP/M favourites.

Unfortunately there is no comparison of products unless you wish to do the job yourself. Nor — and the author confirms this for us — can we rely on the packages having been fully tested. Much information has merely been culled from the manuals and we know how reliable they are!

The book has tried to keep up to date but recent interesting releases such as the Acorn Z80 software, DataGem, Sage, and Systematic, are not included.

Nor can such a book take time off to explain about business practice or give advice on installing a computer system. As it stands it is more a compendium of commercial software, and to be fair, not just an uncritical one.

There is no doubt that a substantial amount of information is packed into the book but I think a few telephone calls to the companies concerned would result in a substantial bundle of sales literature and spec sheets (up to date) and would save you the £7.95.

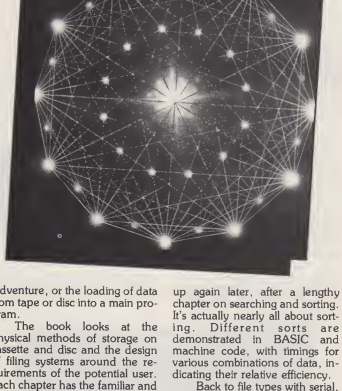
Using the BBC Micro in Education by Don Thorpe. Published by Interface Publications. Price: £5.25.

I have never read a book quite like this before! I started to read, expecting to find the usual mishmash of educational programs, or sadly what is often passed off as educational, with listings of these ready to type in. Instead, I found myself as a teacher being drawn into this book, which attempts to show how a course of study might be set up using a BBC micro, or two. The course is explained in detail, but this is only one of four sections within this book.

The first section is rather sadly called "First steps in a new land", a title which hardly helps to lessen the aura of mystery surrounding micros in schools. I found that this section would be extremely interesting reading matter for a non-computing colleague, but I was also shown a couple of ways to looking at things anew myself! Who says old dogs can't learn new tricks!?

Running a first course in BBC Basic is indeed a challenger. I have attempted to teach some elements to young children, and am currently teaching computer programming at night school, where my class includes a gentleman of 75. This course is aimed fair and square at school, but the general points of consideration are very similar. Some seem quite important to consider, such as the establishing of rules, and the importance of "games", but other questions such as "Should a charge be made for damage?" seem to be beyond any general answer in a book. The explanation of Keywords begins in promising fashion, but I was unable to follow the train of thought which the author followed. Most of the information in this part would perhaps be better gleaned from the excellent User Guide, which is not too difficult for Secondary school pupils.

FILING SYSTEMS AND DATABASES FOR THE BBC MICRO



The third section is entitled "Designing an Educational Program". Here, I feel, the author has assumed that his audience are perfectly conversant with BBC Basic, because little explanation is given as to the use of certain instructions, and yet the same audience is assumed to be incapable of designing a title screen! I found this a rather odd approach. However, Mr. Thorpe redeems himself with a rather nice selection of special effects demonstrations. These provide enough information for the keen teacher to adapt to his own personal purposes.

The final section describes the system employed by the author to set into motion the writing of a program required by a non-computing colleague. I confess I have always steered well clear of this kind of activity, as I generally seem to be working a ten or eleven hour day anyway, what with lesson preparation and marking etc. Should the reader be foolish enough to bite off even more, then this section might perhaps finally put him off! I found this final section out-of-place, and would have been happier if it had been left out.

Overall, this is not a great book, but there is enough of interest and contention to enliven many a staff meeting!

Filing Systems and Databases for the BBC Micro by A.P. Stephenson and D.J. Stephenson. Published by Granada. Price: £7.95.

This time last year the sole database contender for the BBC was Psion's Vufile. It's still a best seller but now just one, and a limited one, such program amongst a sea of software.

While the BetaBases, StarDataBases, DataGems, Indexes (Indices?), Record Keepers, Club Managers etc etc, were being hatched, the Stephenson partnership spotted the trend. They produced a book of great interest to those who use the various aspects of the BBC filing system, in programming or in using one of the many databases, spreadsheets or even save facilities in an Adventure, or the loading of data from tape or disc into a main program.

The book looks at the physical methods of storage on cassette and disc and the design of filing systems around the requirements of the potential user. Each chapter has the familiar and welcome summary of points and self-test section.

Anyone coming new to this area will find a clear description of both the concepts and specific programming methods employed. We start with a simple RAM based sequential file (the RAM based Vu-file remains the best bet for tape users) and build on it, introducing more complex functions, opening and closing files, reserving space, taking orders from a menu, selecting, amending and searching records. All these tasks are coded within PROCedures and FNs and the listings not difficult to follow. A full implementation of a RAM based system is listed and the programming and operation explained.

As a footnote to this program the authors introduce the idea of reserving space on disc for more than one file. This is taken up again later, after a lengthy chapter on searching and sorting. It's actually nearly all about sorting. Different sorts are demonstrated in BASIC and machine code, with timings for various combinations of data, indicating their relative efficiency.

Back to file types with serial, sequential and indexed sequential files. Full examples of a serial and indexed system are listed and the definitions gone into in great detail. At this stage I would have appreciated some shorter examples to type in, to clarify the rules of operation I was reading about, rather than have to tackle another long program. I decided to work out some of my own, but this might put off the beginner.

The final example is of a direct access filing system with fixed length records. The limitations of this system lead to the introduction of hash-coding techniques and tombstone markers. I will leave the authors to explain them.

For me this book greatly improved my understanding of the techniques being employed in the filing system software I use with the BBC. It may also encourage me to have a go myself sometime, to add my personal version to the many already available. The problem with any system is a certain lack of flexibility and personalisation may be the answer. This book will supply the know how, you must supply the task and the time.

Educational Programs for the Electron by Ian Murray. Published by Century Communications Ltd. Price: £6.95.

Ian Murray is a secondary school teacher, and was at a London school when these programs were originally written for the BBC micro. They were tested out in the classroom, and proved to be very successful, according to the author. However, I would imagine that Mr. Murray himself must be writing rather better programs than these nowadays. They give the feel of early, "pioneering" work with a new micro, as the BBC was then, but they now appear to be a little old-hat.

It's not that they are bad programs, some have a fairly good style, but for educational purposes they are not written with much clarity. Also, I would question the educational content of some, although they might therefore appeal to a wider audience!

Quite a few are long, but those with slow fingers will be able to obtain a cassette for a further £6.00. Certainly the long Stockmarket program is well worth typing in, but it does run to about nineteen pages! You have to be dedicated to give your fingertips that kind of bashing.

None of the programs appear to be scaled down for the Electron, but I've not seen the full BBC versions. I would imagine that the BBC listings would make a little more use of sound, for instance, than the new listings in this book.

Overall, I am torn between a great admiration for some of these programs, clearly written and free of errors, and a feeling of surprise as to why others are included in this same volume. I feel that Century might have done better to prune down the number of programs, and to throw in the cassette for free.

Machine Code Made Easy

Name that tune! In keeping with the rest of the mag, Machine Code goes musical.

It is all very well me giving you lists of commands and address modes, but nothing beats an example. This month we look at a simple SOUND routine which I wrote to save space in a long BASIC program and thus prevent the dreaded 'No Room' message.

The routine itself is not very elegant but can easily be modified for your own use. It does however demonstrate many principles which we have looked at in previous articles. It is loaded and set up by a heading program , which subsequently chains the main program, and this assembles the routine and sets up a data store.

Both the routine and the data reside in the area below the normal BASIC program storage space. Thus they don't count in 'room' calculations.

THE THEORY OF THE ROUTINE

As the routine is written, data for tunes is stored as a sequence of pitch and duration values, in this case the data starts at &D80. The routine itself utilises two of the machine code calls provided by Acorn — namely OSBYTE and OSWORD. The OSWORD call requires an eight-byte sequence known as a parameter block for which I have used zero-page space reserved for user routines — &71 to &7A.

Right, so how do we go about writing such a routine? The flowchart in Figure 1 illustrates the sequence which I wanted, and in the following explanation I have indicated which boxes relate to the discussion.

Well the principle is that a number of tunes can be stored, note by note, in sequence. To play them we need to know how far through the data they start and how many notes are to be played. Thus this information must be provided by the BASIC program calling the routine — the INPUT to the routine — box 1.

The parameter block required by the OSWORD call is to provide the same information as a SOUND command — i.e. Channel, Volume, Pitch, Duration. Each of the four is provided by two bytes, one of which is, for our purpose, fixed. So then we must set up this block in a suitable place, providing the four fixed parameters at the beginning and then feeding the variable values required for each note in turn.

The sequence of information required for the block is shown in Figure 2 and is supplied by box 2 for the fixed values and box 3 as each note is passed in the loop for playing the tune. We have now entered the tune playing loop!

Having collected the information required for a note in the parameter block we now check to see if the sound buffer is full. This is achieved by using one of the OSBYTE calls, namely that with the accumulator containing &80 osword. The osword call we test again, repeating this process until there is free space in it — thus we loop around box 4 in the flowchart until a space occurs.

Once the buffer has room we send a note to channel 1 using the OSWORD parameter block and call, then the volume, pitch and channel are changed to give a second note. This is two octaves higher than the first to give a harmony effect as the two will play all but simultaneously — machine code being so fast! This note is in effect sent to the buffer for channel 2 by calling OSWORD again and the delay between the two is in the order of 20 microseconds, hardly an audible delay.

Having sent the note to both channels we are not ready to get the second note from the data store! Thus the offset counter is incremented to the next set of data and the count of notes to be played is decremented. If the count is not zero there are still notes to be played, so we test the count and go back for another note if required. Otherwise the routine ends and returns control to the main program.

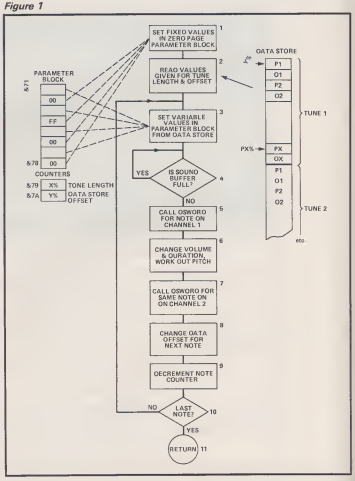
THE ROUTINE — BASIC/ ASSEMBLER PROGRAM

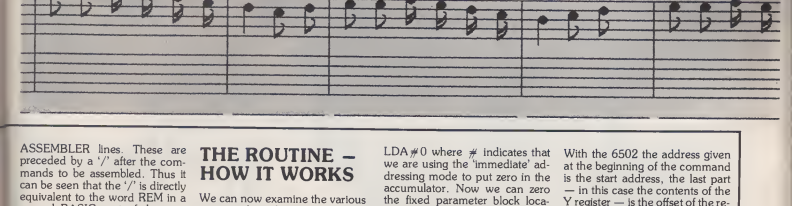
The program that I have given here is part of a program that would display game titles, set up character definitions etc before chaining the main program. It first creates the 'data store' by reading DATA and then putting in into locations &D80 onward. This is followed by the ASSEMBLER part, commencing at line 100 and this we will examine in more detail in a moment.

First, however, note two things in this program. In line 110 we load P% with the value &D00 — on entering the ASSEMBLER we must tell it where to assemble the code and the value of P% on entry gives the start address (we will examine this more fully next month). Forgetting to set P% means that your code could end up almost anywhere!

Secondly note the comments that I have added in the

Figure 1





ASSEMBLER lines. These are preceded by a '/' after the commands to be assembled. Thus it can be seen that the '/' is directly equivalent to the word REM in a normal BASIC part of the program. These comments need not be typed in and are there solely for your instruction as to how the commands operate.

Look through the program now and examine how it works. After the listing we discuss each part of the code more fully.

THE ROUTINE — HOW IT WORKS

We can now examine the various commands and how they work. This is probably best done line by line.

LINE 120 — the 6502 has no individual command which will clear either the accumulator or any memory location. Thus to zero any location we have to load it with the number zero, hence LDA#0 where # indicates that we are using the 'immediate' addressing mode to put zero in the accumulator. Now we can zero the fixed parameter block locations of the next three commands by copying the accumulator zero into each in turn — STA&72 etc.

The assembler would allow us to use STA&0072 but this is absolute addressing not zero-page addressing, it is unnecessary, uses another byte and is slower — always use zero-page mode if possible. We then load the accumulator with 255 (LDA#&FF) and use it to copy 255 into location &74 — in the parameter block.

LINE 130 — whenever we wish to call the routine we must give it the number of notes to play and where to start picking up the data. In calling a BASIC program it is possible to pass parameters into the machine code routine by using A%, X% or Y%. When a call is made the accumulator, X register and Y register contain the least significant byte of the last respective values given to the above resident variables.

In this case the calling program would be: X% = 20: Y% = 0: CALL%D00 (see line 240), thus in line 130 we pick up the values of X% (X register) and Y% (Y register) and store them in our counter stores at &79 and &7A respectively.

A second tune could be stored immediately after the first — all that is required to play it would be a new value of X% and Y% when the routine is called.

LINE 140 — The channel and volume are selected to 1 and −15 respectively and then passes to their correct places in the parameter block and we are now in the beginning of the loop (box 3). Note how we achieve a negative number by the use of &F1 (&FF= −1, &FE= −2 etc). However, after the four bytes that achieve this we now come to the collection of the data from the store. This is a frequently encountered function, namely the selection of one or more bytes from a 'look up' table.

It is achieved in nearly all microprocessors by one or more forms of 'Indexed Addressing'. With the 6502 the address given at the beginning of the command is the start address, the last part — in this case the contents of the Y register — is the offset of the required address from the start. Thus the &D80 is always the start address and Y is loaded with the offset which Y was given at entry to the routine.

At the end of the loop the location &7A is incremented by two. So on the next turn around the loop the offset loaded into Y from &7A will pick up the next address.

LINE 150 — At the beginning of this line is 'check'. This is a line label used by the assembler as a reference point for jumps and branches (we will look in more detail next month). Following this the X, Y and accumulator registers are loaded with specific values before a call is made to the famous OSBYTE call at &FFF4. For the time being all we are interested in is that A = %80 (128) and that this call is equivalent to *FX128 in BASIC.

The purpose of the call is to check the contents of Sound channel 1 in this case, and it returns the number of spaces left in the buffer in register X. If the contents of the X register is transferred into the accumulator then the flag register is set. Thus for no spare spaces (returning 0 in X) then the zero flag will be set by executing the command TXA.

Then by using the BEQ branch comparison, which means if the zero flag is set, then branch to. . . . , we arrange to try the test again until the buffer does have room. This is rather like dialling a phone number repeatedly until it is not engaged! 'BEQ check' says — branch to the line with the label 'check' if the zero flag is set.

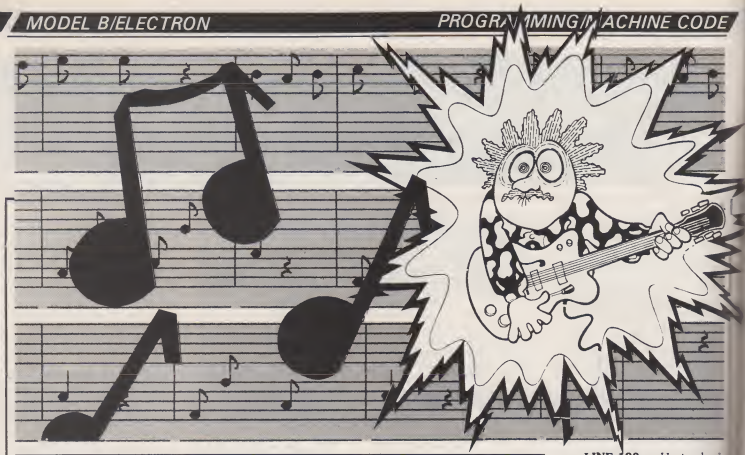
LINE 160 — Having found space in the Sound buffer, we will now try and fill it by sending a new note to it. This is achieved by calling the routine OSWORD in the operating system. In this case, what I will call OSWORD7 (i.e. with 7 in the accumulator).

As you know we have the required SOUND information now stored in our parameter

CONTINUED OVER

PROGRAM LISTING

```
   10 REM ------------------------
20 REM READ AND SAVE NOTES FOR MUSIC
30 REM ------------------------
40 FOR X% = 0 TO 19 : READ P%,D% :
   ?(&D80+X%*2)=P% : ?(&D80+X%*2+1)=D% : NEXT
50 DATA 69,16,69,4,73,4,73,12,69,2,73,8,69,2,
61, 8,53,2,49,16,81,12,73,12,69,2,53,2,73,2,8
1,6,73, 6,69,12,61,12,53,12
70 REM ------------------------
80 REM ASSEMBLE MACHINE CODE
90 REM ------------------------
100 FOR I% = 0 TO 3 STEP 3
110 P%=&D00
120 [ OPT I% : LDA#0 : STA&72 : STA&76 : STA&7
8 : LDA#&FF : STA&74  /   set the fixed param
eters of the OSWORD parameter block in zero-p
age
130 STX&79 : STY&7A  /  collect the tune leng
th and the data offset from X% & Y% respectiv
ely
140 .start LDA#1 : STA&71 : LDA#&F1 : STA&73
: LDY&7A : LDA&D80,Y : STA&75 : LDA&D81,Y : S
TA&77  /  store the variable parameters in th
e block including collecting the first note f
rom the data store
150 .check LDY#&FF : LDX#&FA : LDA#&80 : JSR&
FFF4 : TXA : BEQ check  /  check the sound bu
ffer to see if there is space in it, check ag
ain until there is
160 LDX#&71 : LDY#0 : LDA#7 : JSR&FFF1  /  se
t X & Y registers to indicate the parameter b
lock and call OSWORD with A=7 — the SOUND com
mand - feeds a note to channel 1
170 LDA#&F6 : STA&73 : LDA#2 : STA&71  /  cha
nge the volume to -10 and the channel to 2
180 LDY#0 : LDX#&71 : CLC : LDA&75 : ADC#&0
1 : STA&75 : LDA#7 : JSR&FFF1  /  reset to the
parameter block and add 2 octaves (96) to the
 pitch, then feed the note to channel 2
190 INC&7A : INC&7A : DEC&79 : BEQend : JMP s
tart  /  increment the data pointer to the ne
xt note data — 2 bytes forward — and decremen
t the number of notes to be played - if zero
finish else go and do another note
200 .end RTS
210 ]
220 NEXT
230 MODE7 : PRINTTAB(5,10)"RECOGNISE THE TUNE
 ?"
240 X%=20 : Y%=0 : CALL&D00
250 END
```



```
OSWORD call with A=7 (SOUND) requires  8 byte parameter block.  Address of
start of block given by Y (hi) and X (lo) on entry.
e.g. X=&71, Y=&0  for &0071

PARAMETERS

Address chosen          Function        Byte            Value

&71                     Channel         LSB             0-4
&72                     Channel         MSB             &00
&73                     Amplitude       LSB             &FF-&F1
&74                     Amplitude       MSB             &FF
&75                     Pitch           LSB             &00-&FF
&76                     Pitch           MSB             &00
&77                     Duration        LSB             &00-&FF
&78                     Duration        MSB             &00

OSBYTE call with A=128 (BUFFER CHECK)

Y=&FF
X=&FB           channel 0
X=&FA           channel 1
X=&F9           channel 2
X=&F8           channel 3
On return X equals the number of free spaces in the channel buffer.

Other values of X cause ADC and other machine buffers to be tested.
```

Fig 2. OSBYTE and OSWORD calls used.

block starting at address &71. We can tell the OSWORD system this by loading the X and Y registers before the call is made – Y contains the high two bytes of the address, X the lower two bytes. Thus we make Y = 0 and X = &71 and then make the call which will make the first note play on channel one.

LINE 170 – To give a harmony effect we now change the volume and channel information stored in our parameter block. This is achieved simply by loading the accumulator with each of the required values in turn (LDA#) and storing them in the required memory location (STA&).

LINE 180 – We achieve the harmony by picking a note two octaves (96 semitones) higher than the main channel 1 note. Thus as we go to call the OSWORD7 routine again we clear the carry flag (CLC), load the contents of our pitch location (at &75) to the accumulator, then add 96 (ADC#&60) and put the resulting value back at &75 in our parameter block. Then it's 'Play it again Sam!' on the second channel.

LINE 190 – Having loaded both the channel 1 & 2 buffers with the note, they will play almost simultaneously as I mentioned above. So we now move the offset pointer to the data to the next note and then reduce the number of notes to be played (DEC&79). If this reduces to zero then the zero flag will again be set.

If the flag is set then there are no more notes to play so a branch is made to the line labelled 'end' (200) which returns control to the main program. Otherwise that returns control to the main program. Otherwise the next instruction JuMPs us back to the 'start' line for the next note.

LINE 200 – This line labelled 'end' is purely the return from subroutine (RTS) that bounces us back to the BASIC program.

Finally the tune is played for your amusement – assuming you loaded it in properly!

Hopefully this little program will encourage you to play around with short tunes. Then perhaps with the routine itself to load say all channels – there is nothing stopping you defining more information by using the data store and then loading up the variable parameters in the block individually.

Having played a little with the assembler, next month we will look at it in more detail. We will introduce you more to the flexibility and the various ways of defining machine code (P% to remember).

Music Machine

R.K. Reading

Make music with the micro maestro.

Musical Machine is primarily a teaching aid for people learning a musical instrument like the flute. The program enables you to enter a tune you want to be able to play directly from the sheet music and then make the computer store it as a file, edit it or play it a number of times. This enables the instrument player to hear how his tune should be played. Also the program can be used to play the second part of a duet or can be used to write your own music.

To use the program first load "musica". This contains the instructions for the main program that follows. This program is automatically loaded when the instructions have been read. The program operates at its best if there is a disk drive, for this allows quick and easy access but it will work on tape as long as you remember whereabouts on the tape your files are situated.

LIST OF VARIABLES

DIM N(X) For the different types of note.
DIM R(X) For the different types of rest.
DIM V(X) For the value of each note.
DIM L(X) For the length of each note.
DIM P(X) For the length of rest between the notes.

LIST OF PROCEDURES

PROCload Loads a tune on tape/disk.
PROCsave Saves a tune on tape/disk.
PROCplay Enables you to hear a tune.
PROCmake Enables you to enter music.
PROCedit Enables you to change entered music.

HOW THE PROGRAM WORKS

The first problem I encountered when formulating my ideas for the program was how the information was to be entered for each note. Was the music going to be entered when playing a tune on the computer or from sheet music? From my own use of such programs I rejected the first idea, on the grounds that one could not make very good tunes. Therefore I opted for the second which allows proper and interesting playback.

The next decision was how to enter the music. It became obvious that it would have to be a note and each note would have different variables like the length l(x), the gap between notes p(x) and the different pitches of the notes v(x). I decided to store these in arrays and therefore every note had a pitch length, value length and gap length.

I then realised that the length of notes and rests varies depending on the time signature of the tune and therefore I had to set the length of the different types of note or rest according to the time signature. This is done in line 900-970.

The best way to understand how the program works is to look at the sound statement. From this you can see that you need at least two variables, the pitch of the note and the length.

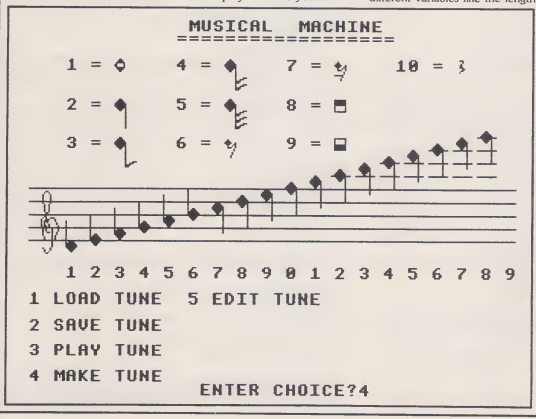
You know what you want to end up with, but if you are entering notes from music you have to work out their pitch and length in different stages because the note could have a sharp or a flat or it could be dotted. This is taken into consideration after the type of note is given its original length and pitch values.

So first you enter the time signature and from this the computer sets the length the note or rest should be played for. Then it will ask questions about the note; if it is dotted it will add more to the original note value, if it is a sharp it will add more pitch value and if the note is a flat it will deduct a bit from the pitch value.

Thus the program works by first setting the initial values and then adding or subtracting, depending on the factors of the particular note.

If a note is slurred a gap between it and the next must be a zero. Therefore the program has to incorporate a delay loop which will be long or short depending on the note being slurred or not. This is done at line 1400 by changing the variable p(x).

The original pitch value of the note is created by choosing a note from the ascending order on the screen and entering 1 to 19. From this number the computer allocates the correct pitch by reading it from data. If you enter 19, which means rest, the computer cannot sound a note that does not exist so it increases the gap from the last note entered by adding to p(x) of the previous note.



The hard part of the program was setting the right lengths for the different notes in each time signature. If you want to insert an envelope into the program it may cause the program to speed up or slow down. To counteract this the variables in lines 900-970 would have to be changed.

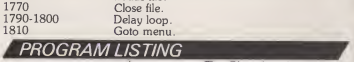
MAIN PARTS OF THE PROGRAM

10-760	Sets up screen display.
859-1320	Allows you to enter tune.
1340-1430	Plays the tune.
1440-1490	Deals with entry of rest.
1510-1650	Saves tune.
1670-1810	Loads tune.

PROGRAM DETAILS

10-40	Notes about program.
50	Sets mode.
70-100	Sets variables.
120-300	Defines user defined characters.
320-440	Print text graphics on screen.
450	Joins text cursor to graphics cursor.
460-490	Print text at graphic locations.
500-560	Print notes in ascending order.
570-600	Draws staves from data.
610-620	Data from staves.
630-680	Draws ledger lines from data.
690	Data for ledger lines.
700	Sets up a text window.
730-760	Prints menu on screen.
770	Accepts choice.
780-820	Depending on choice does procs.
860-870	Accepts time signature.
880-890	Checks for illegal entry.
900-970	Depending on time signature sets up length of notes and rests.
980	Accepts note number.
990-1000	Checks for illegal entry.
1010	Accepts note value if 19 goto procrest.
1020-1030	Checks for illegal entry.
1040-1060	Gives pitch value to note entered.
1070	Data for pitch value.
1080	Accepts if note has flat or sharp.
1090-1100	Checks for illegal entry.
1110-1120	Increase or decrease pitch value accordingly.
1130	Accepts type of note.
1140-1150	Checks for illegal entry.
1160	Accepts if dotted note.
1170-1180	Check for illegal entry.
1190-1200	Increase length value of the note accordingly.
1210	Accepts if slurred.
1220-1230	Checks for illegal entry.
1240	If slurred then makes gap between notes.
1250	Accepts enter edit any more notes.
1260-1270	Checks for illegal entry.
1280	If answer yes then goto 980.
1290	If answer no accepts no. of notes in tune.
1300-1310	Checks for illegal entry.
1320	Goto menu.
1350	Accepts how many times tune to be played.
1360-1370	Checks for illegal entry.
1380	Sets loop for times to be played.
1390	Sets loop for no. of notes in tune.
1400	Produces gap between notes.
1410	Plays note.
1420	Loops: stops sound after last note.
1430	Goto menu.
1440-1480	Accepts rest no. goto 1010.
1510	Save tune procedure.
1520	Accepts file name.
1530	Check file name not longer than three letters.
1570-1600	Saves variables into file.
1610	Closes file.
1630-1640	Delay loop.
1650	Goto menu.
1670	Load file procedure.
1680	Accepts file name.
1690	Check file name not longer than three letters.
1730-1760	Loads file.
1770	Close file.
1790-1800	Delay loop.
1810	Goto menu.

PROGRAM LISTING

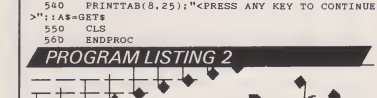


```
   50   MODE4
   60   CLS
   70   VDU23;8202;0;0;0;
   80   VDU19,1,6,0,0,0,19,2,4,0,0,0
   90   COLOUR2:COLOUR129:PRINT:SPC(50);"INSTRUCTI
ONS":SPC(18)
  100   COLOUR3:COLOUR128
  110   PRINT:PRINT
  120   PRINT"This program contains the instructio
ns"
  130   PRINT"for the music program that follows.W
hen"
  140   PRINT"the next program has been loaded and
 is"
  150   PRINT"running the first thing you will be"
  160   PRINT"confronted with is a screen display"
  170   PRINT"On the screen will be a sereis of no
tes"
  180   PRINT"in ascending order with a code numbe
r"
  190   PRINT"under them.Above these notes will be
 the"
  200   PRINT"different type of note which you can
"
  210   PRINT"choose followed by there code number
"
  220   PRINT"The computer will first ask you to e
nter"
  230   PRINT"1 LOAD TUNE,2 SAVE TUNE,3 PLAY TUNE"
  240   PRINT"4 MAKE TUNE,5 EDIT TUNE and you repl
y"
  250   PRINT"with the number you want.When making
 a"
  260   PRINT"tune the computer will first ask"
```

CONTINUED OVER

```
  270    PRINT"NUMBER OF THE NOTE this means you en
ter"
  280    PRINT"1 if it is the first note in the tun
e"
  290    PROCWAIT
  300    PRINT"Next it will ask you to enter the CO
DE"
  310    PRINT"NUMBER OF THE NOTE this being the nu
mber"
  320    PRINT"directly below the note which is in
the"
  330    PRINT"ascending order of notes.Next it wil
l"
  340    PRINT"ask DOES IT HAVE A SHARP OR FLAT the
 "
  350    PRINT"answer being 1 for sharp,2 for flat
and"
  360    PRINT"RETURN for neither.It will then ask
for"
  370    PRINT"THE TYPE OF NOTE it is i.e. the code
"
  380    PRINT"number which refers to.for example,a
"
  390    PRINT"crotchet.Next is it a DOTTED NOTE an
swer"
  400    PRINT"y or n.Next is it SLURRED answer y o
r n."
  410    PRINT"And last of all it will ask if you w
ant"
  420    PRINT"to enter more notes or edit any answ
er"
  430    PRINT"y or n.If there is a rest you enter
0"
  440    PRINT"when it asks for the note number"
  450    PRINT"then enter the type of rest it is."
  460    PRINT"It won't except a file name greater"
  470    PRINT"than three letters long.To edit"
  480    PRINT"you enter the number of the note"
  490    PRINT"you want to edit and follow the"
  500    PRINT"normal procedure for entering a new
tune"
  510    PROCWAIT
  520    CLS:PRINTTAB(8,20);"LOADING MUSICAL MACHIN
E":PRINTTAB(15,21);"":CHAIN"MUSICB"
  530    DEF PROCWAIT
  540    PRINTTAB(8,25);"<PRESS ANY KEY TO CONTINUE
>":A$=GET$
  550    CLS
  560    ENDPROC
```

PROGRAM LISTING 2



```
   50    MODE4:CLS
   60    REM SET VARIABLES
   70    DIM N(5):DIM R(10):DIM V(200):DIM L(200):D
IM P(200)
   80    FOR X= 1TO 200
   90    LET P(X)=1
  100    NEXT
  110    REM DEFINE CHARACTERS
  120    VDU 23,240,24,60,126,255,255,126,60,24
  130    VDU 23,241,24,60,126,194,194,126,60,24
  140    VDU 23,242,0,32,112,249,115,37,57,2
  150    VDU 23,243,255,255,255,255,129,129,129,255

  160    VDU 23,244,255,129,129,129,255,255,255,255

  170    VDU23,245,0,24,36,66,66,66,66,66
  180    VDU23,246,66,36,36,36,36,36,24,16
  190    VDU23,247,48,83,156,184,168,164,180,147
  200    VDU 23,248,0,128,64,64,32,32,16,16
  210    VDU23,249,81,81,77,65,65,33,33,31
  220    VDU23,250,16,16,32,32,64,64,64,192
  230    VDU23,251,1,1,1,1,1,66,34,36,24
  240    VDU23,252,128,128,128,128,128,128,128,128

  250    VDU23,253,128,128,128,128,128,155,156,240,224
  260    VDU23,254,135,156,240,224,135,156,240,224

  270    VDU23,226,56,8,16,48,56,12,24,32
  280    VDU23,255,128,128,128,128,128,128,128,128
  290    VDU23,227,2,4,4,8,8,0,0,0
  300    VDU23,225,2,252,4,8,8,0,0,0
  310    REM SCREEN DISPLAY
  320    PRINTTAB(13,2);"MUSICAL  MACHINE";
  330    PRINTTAB(12,3);"==================";
  340    PRINTTAB(3,5);"1 = ";CHR$(241);"    4 = ";
CHR$(240);"    7 = ";CHR$(242);"    10 = ";CHR$(22
6)
  350    PRINTTAB(25,6);CHR$(225)
  360    PRINTTAB(16,12);CHR$(227)
  370    PRINTTAB(3,8);"2 = ";CHR$(240);"    5 = ";
CHR$(240);"    8 = ";CHR$(243)
  380    PRINTTAB(3,11);"3 = ";CHR$(240);"    6 = "
;CHR$(242);"    9 = ";CHR$(244)
  390    PRINTTAB(3,21);"1 2 3 4 5 6 7 8 9 0 1 2 3
4 5 6 7 8 9";
  400    PRINTTAB(1,15);CHR$(245);
  410    PRINTTAB(1,16);CHR$(246);
  420    PRINTTAB(1,17);CHR$(247);CHR$(248);
  430    PRINTTAB(1,18);CHR$(249);CHR$(250);
  440    PRINTTAB(1,19);CHR$(251);
  450    VDU5
  460    MOVE255,752:PRINTCHR$(252):MOVE543,848:PRI
NTCHR$(252)
  470    MOVE255,720:PRINTCHR$(252):MOVE543,816:PRI
NTCHR$(254)
  480    MOVE543,752:PRINTCHR$(253):MOVE543,720:PRI
NTCHR$(254)
  490    MOVE254,656:PRINTCHR$(252):MOVE254,624:PRI
NTCHR$(253)
  500    LET X=96:LET Y=416
  510    FOR T=1 TO 18
  520    MOVEX,Y:PRINTCHR$(240)
  530    IF T>=7 THEN MOVEX+31,Y-16:PRINTCHR$(252):
MOVEX+31,Y-48:PRINTCHR$(252):GOTO550
  540    MOVE X,Y+16:PRINTCHR$(255):MOVE X,Y+48:PRI
NTCHR$(255)
  550    LET X=X+64:LET Y=Y+16
  560    NEXT
  570    FOR X=1 TO  5
  580    READ A:READ B:READ C:READ D
  590    MOVE A,B:DRAW C,D
  600    NEXT
  610    DATA 0,416,1280,416,0,448,1280,448,0,480,1
280,480
  620    DATA 0,512,1280,512,0,544,1280,544
  630    Y=576
  640    FOR P=1 TO 16
  650    READX
  660    MOVEX,Y:DRAW X+48,Y
```

```
  670   IF P=7 OR P=12 OR P=15 THEN LET Y-Y+32
  680   NEXT
  690   DATA 795.859,923,987,1051,1115,1179,923,98
7,1051,1115,1179,1051,1115,1179,1179
  700   VDU4
  710   REM PRINT COMMANDS
  720   VDU28,0,31,39,23
  730   PRINT"1 LOAD TUNE  5 EDIT TUNE":VDU10
  740   PRINT"2 SAVE TUNE":VDU10
  750   PRINT"3 PLAY TUNE":VDU10
  760   PRINT"4 MAKE TUNE"
  770   PRINTTAB(20,28);"                    ENTER CHOIC
E";:INPUT J
  780   IF J=1 THEN PROCLOAD
  790   IF J=2 THEN PROCSAVE
  800   IF J=3 THEN PROCPLAY
  810   IF J=4 THEN PROCMAKE
  820   IF J=5 THEN PROCMAKE
  830   GOTO 730
  840   REM ENTER TUNE
  850   DEFPROCMAKE:CLS
  860   PRINT"ENTER TIME SIGNATURE TOP NO. FIRST"

  870   INPUT A:INPUT B
  880   IF A>1 AND A<7 AND B>3 AND B<9 THEN GOTO 9
00
  890   CLS:GOTO 860
  900   IF A=6 AND B=8 THEN LET N(1)=10.5:N(2)=5.2
5:N(3)=2.625:N(4)=1.3125:N(5)=0.65625
  910   IF A=4 AND B=4 THEN LET N(1)=14:N(2)=7:N(3
)=3.5:N(4)=1.75:N(5)=0.875
  920   IF A=3 AND B=4 THEN LET N(1)=10:N(2)=5:N(3
)=2.5:N(4)=1.25:N(5)=0.625
  930   IF A=2 AND B=4 THEN LET N(1)=8:N(2)=4:N(3)
=2:N(4)=1:N(5)=0.5
  940   IF A=6 AND B=8 THEN LET R(6)=2.625:R(7)=1.
3125:R(8)=21:R(9)=10.5:R(10)=5.25
  950   IF A=4 AND B=4 THEN LET R(6)=3.5:R(7)=1.75
:R(8)=28:R(9)=14:R(10)=7
  960   IF A=3 AND B=4 THEN LET R(6)=2.5:R(7)=1.25
:R(8)=20:R(9)=10:R(10)=5
  970   IF A=2 AND B=4 THEN LET R(6)=2:R(7)=1:R(8)
=16:R(9)=8:R(10)=4
  980   CLS:PRINT"ENTER NOTE NO.":INPUTX:CLS
  990   IF X>0 THEN GOTO 1010
 1000   GOTO 980
 1010   PRINT"ENTER NOTE VALUE":INPUT C:IF C=19 TH
EN PROCREST:GOTO1250
 1020   IF C>0 AND C<20 THEN GOTO 1040
 1030   CLS:GOTO 1010
 1040   FOR T=1 TO C
 1050   READ D:NEXT
 1060   LET V(X)=D
 1070   DATA109,117,121,129,137,145,149,157,165,16
9,177,185,193,197,205,213,217
 1080   CLS:PRINT"DOES IT HAVE A SHARP OR FLAT 1 O
R 2":INPUT Z
 1090   IF Z=1 OR Z=2 OR Z=3 THEN GOTO 1110
 1100   GOTO 1080
 1110   IF Z=1 THEN LET V(X)=V(X)+4
 1120   IF Z=2 THEN LET V(X)=V(X)-4
 1130   CLS:PRINT"ENTER TYPE OF NOTE":INPUT E
 1140   IF E>0 AND E<6 THEN GOTO 1160
 1150   GOTO 1130
 1160   CLS:PRINT"IS IT A DOTTED NOTE":INPUT C$
 1170   IF C$="N" OR C$="Y"THEN GOTO 1190
 1180   GOTO 1160
 1190   IF C$="Y"THEN LET L(X)=N(E)+(N(E)/2)
 1200   IF C$="N"THEN LET L(X)=N(E)
 1210   CLS:PRINT"IS IT SLURRED":INPUTC$
 1220   IF C$="N" OR C$="Y"THEN GOTO 1240
 1230   GOTO 1210
 1240   IF C$="Y"THEN LET P(X)=0 ELSE P(X)=1
 1250   CLS:PRINT"ENTER OR EDIT MORE NOTES":INPUT
C$
 1260   IF C$="N" OR C$="Y"THEN GOTO 1280
 1270   GOTO 1250
 1280   IF C$="Y"THEN RESTORE1070:GOTO 980
 1290   CLS:PRINT"ENTER AMOUNT OF NOTES IN TUNE":I
NPUTQ
 1300   IF Q>0 THEN GOTO 1320
 1310   GOTO 1290
 1320   CLS:RESTORE1070:ENDPROC
 1330   REM TUNE
 1340   DEFPROCPLAY
 1350   CLS:PRINT"HOW MANY TIMES":INPUTC
 1360   IF C>0 THEN GOTO 1380
 1370   GOTO 1350
 1380   FOR T=1 TO C
 1390   FOR Y=1 TO Q
 1400   SOUND &0001,0,0,P(Y)
 1410   SOUND &0001,-15,V(Y),L(Y)
 1420   NEXT:NEXT:SOUND &0001,0,0,0:CLS
 1430   ENDPROC
 1440   DEFPROCREST
 1450   CLS:PRINT"ENTER REST NUMBER":INPUT C
 1460   IF C>5 AND C<11 THEN GOTO 1480
 1470   GOTO 1450
 1480   LET P(X)=1+R(C)
 1490   ENDPROC
 1500   REM SAVE FILE
 1510   DEFPROCSAVE
 1520   CLS:PRINT"ENTER FILE NAME":INPUT FILE$
 1530   LET FILE=INT(VAL(FILE$)):IF LEN(FILE$)>3 T
HEN PRINT"FILE NAME TOO LONG":GOTO1390
 1540   FILE$="FILE"+FILE$
 1550   Y=OPENOUT FILE$
 1560   PRINT"SAVING"
 1570   PRINT#Y,X,Q:FOR S=1 TO Q
 1580   PRINT#Y,V(S),L(S),P(S)
 1590   NEXTS
 1600   PRINT#Y,A,B
 1610   CLOSE#Y
 1620   CLS:PRINT"FILE HAS BEEN SAVED"
 1630   FOR V=1 TO 1000
 1640   NEXT
 1650   ENDPROC
 1660   REM LOAD FILE
 1670   DEFPROCLOAD
 1680   CLS:PRINT"ENTER FILE NAME";:INPUT FILE$
 1690   LET FILE=INT(VAL(FILE$)):IF LEN(FILE$)>3 T
HEN GOTO 1680
 1700   FILE$="FILE"+FILE$
 1710   Y=OPENIN FILE$
 1720   PRINT"LOADING";FILE$
 1730   INPUT#Y,X,Q:FOR S=1 TO Q
 1740   INPUT#Y,V(S),L(S),P(S)
 1750   NEXTS
 1760   INPUT#Y,A,B
 1770   CLOSE#Y
 1780   CLS:PRINT"FILE HAS BEEN LOADED"
 1790   FOR V=1 TO 1000
 1800   NEXT
 1810   ENDPROC
```

A & B'S NATIONWIDE DEALERGUIDE 01 - 437 0699

CHESHIRE

FAIRHURST INSTRUMENTS LTD

Complete range of BBC equipment including Econets, Printers, Plotters, Colour Monitors, Graphic Tablets. Upgrades. Disc Drives. Disc Controller chips. Torch Computers. Z80 Discpack. Extensive range of Software

Dean Court, Woodford Rd., Wilmslow, Cheshire Tel: 0625 533741

NORTHERN COMPUTERS

Churchfield Road, Frodsham, Cheshire WA6 6RD. Telephone: (0928) 35110

Supply and service for Electron, BBC, Econet, Apple Apricot, IBM, ICI Rexagon interfaces. Exclusive Amstrad educational distributors. Design & manufacture of MICROPULSE products incl. BBC/Spectrum disc drives, the BBC external ROM Box, Disc-Smith bit copy & repair utility, Micropulse Gold discs, Youngtrainer robotics trainer & buggy. For information send SAE. Factory shop open 9 am — 5.30 pm Mon-Sat

ESSEX

ESTUARY

SOFTWARE PRODUCTS

Estuary now have BBC's in stock together with a wide range of software and accessories. Complete spares kit in stock. The home computer centre.

261 Victoria Ave., Southend-on-Sea. Phone: (0702) 343568

KENT

MEDWAY COMPUTERS LTD.

• BBC authorized dealers and service centre • Torch Z80 disc pack available • Our own credit facilities • Access and Barclaycard welcome •

We are open 9 till 5.30 six days a week

141 New Rd., Chatham, Kent ME4 4PT. TEL: (0634) 826080

LANCASHIRE

HCCS

MICROCOMPUTERS
120/122 Darwen Street, Blackburn, Lancs Tel: 0254 672214

Open 9am to 5.30pm Mon-Sat (except Thurs 9am to 12.30)

ACORN/BBC DISTRIBUTOR

Forth and Pascal for BBC Micro and for Epson HX20. Printers, Disc Drives, Consumables.

LEICESTERSHIRE

Leigh Computer Systems

NEVER KNOWINGLY UNDERSOLD

Official Acorn/BBC dealer and service centre
7 Coventry Road, Hinckley, Leics LE10 1QF Tel: 0455 612139

BBC Model B, Electron Disk drives – Cumana Disc drives from CSX £149. Phone for prices larger disc drives Shugart 100K singles, BBC discs, TV (monitors, accessories & software). Printers – Epson FX80, RX80 and many more. Also Sinclair agents, Oric agents and Dragon service centre

MICRO-MAYS

OFFICIAL ACORN/BBC DEALER

BBC Model B, Electron **Printers:** Epson FX80, RX80, Shinwa CP80, Seikosha GP100, Star DP510. **Disk Drives:** Teac 100K single, twin, Shugart 100K singles BBC Disks, TV (monitors, accessories and software). Also agents for Atari, Commodore, Sinclair, Oric and Dragon

MAYS COMPUTERS, 57 Churchgate, City Centre, Leicester LE1 3AL (0533) 22212

LONDON

THE VIDEO PALACE

Londons largest home computer store. Model B and Torch. ZX and Commodore 64 stockists. Full range of games software.

100 Oxford Street, London W1 Tel: 01-637 0366

PAUL ELECTRICAL LTD

250-252 Grand Drive, Raynes Park, SW20. Tel: 01-542 6546

Official Acorn dealer. Full range of Cumana and Disc Drives in Stock

ALSO TRADING AS

Woods Radio, 257 Lavender Hill, SW11
01-228 2682

Supply and Repairs to Education and Local Councils

GREATER MANCHESTER

Leigh Computer Systems

NEVER KNOWINGLY UNDERSOLD

Official Acorn/BBC dealer and service centre **75 Cross Street, Sale**. BBC Model B, Electron, Disk Drives — Comana disc drives from CSX £139. Phone for prices of larger disc drives. Shugart 100K singles, BBC discs. TV (monitor accessories & software). Printers — Epson FX80, RX80 and many more. Also Sinclair agents, Oric agents and the Dragon service centre.

MIDDLESEX

TWICKENHAM COMPUTER CENTRE

Acorn • BBC • distributors and Apricot dealers

Micro Computers for home and business plus peripherals/software and accessories. Always a wide range in stock at

72 Heath Road, Twickenham, Middx. TEL: 01-892 7896

MERSEYSIDE

MICROMAN Computers

ACORN STOCKISTS & SERVICE CENTRE
Complete range of Acorn/BBC equipment & upgrades. Printers (Star, Epson, Juki). Disc Drives (Pace, Torch Z80). Specialist ROM's & peripheral equipment (Solidisk, Computer Concepts, Educational Software etc.)

Rainford Industrial Estate, Mill Lane, Rainford, St Helens, Merseyside. Tel: (074488) 5242

WEST SCOTLAND

WEST COAST PERSONAL COMPUTERS

BBC, Acorn and Torch dealers. Range of Disk Drives, Printers and Monitors on display

47 Kyle Street, Ayr. (0292) 285082

SHETLAND ISLES

We stock a wide range of software books and peripherals

Tomorrows World

Local service and northern mail order centre

20 Commercial Road, Lerwick, Shetland Isles (0595) 2145 B B C

SUFFOLK

Suffolk Computer Centre

BBC Microcomputer Service & Information Centre

Microcomputers • Disc Drives • Monitors Matrix & Daisywheel Printers • Joysticks Cassettes • Light Pens • Graphics Tablet Books & Software

3 Garland St., Bury St Edmunds. Telephone: 0284 - 705503
Open: Mon - Sat 9 - 5.30

SURREY

SIMNETT COMPUTERS LIMITED

One of the UK's largest independent suppliers of microcomputer equipment. Ring **01-541 1495** or visit Unit 14, St George's Ind. Est. 380 Richmond Road, Kingston upon Thames, Surrey KT2 5QB

BUYING GROUP — YES THAT'S US!

CROYDON COMPUTER CENTRE

CCC

Official Acorn dealer and service centre. Full range of peripherals and spares for BBC Micro Electron, Torch etc

29A Brigstock Rd, Thornton Heath, Surrey.

BRING THIS COUPON FOR £5 DISCOUNT

Tel: 01 - 689 1280

SURREY

STATACOM

18 & 20 Grove Road, Sutton, Surrey. Tel: 01-661 2266

Computers and Peripherals
3" Hitachi Drive 5¼" range
Printers and Monitors
2nd processors, business machines.

A & B'S NATIONWIDE DEALERGUIDE
01 - 437 0699

SUSSEX

BBC B's Plus Range of Printers/Disc Drives/ Monitors. On Site Servicing/Upgrades – Variety of Software Courses on BBC Micro from £15.00

michael
Business Systems Ltd
195 London Rd., Burgess Hill, Sx. Tel: 04446 45636

WALES

GWENT COMPUTERS
Everything for the BBC computer.
ACORN AUTHORISED DEALERS
CUMANA DISTRIBUTOR
92 Chepstow Road, Newport NP9 8EE
Tel: 0633 841760

YORKSHIRE

MICRO POWER LTD.
The leading B.B.C. dealer in the North
NORTHWOOD HOUSE
NORTH STREET LEEDS LS7 2AA
TEL. LEEDS (0532) 458800

TYNE AND WEAR

HCCS
533 Durham Road, Low Fell, Gateshead
TEL: Newcastle 091 487 2469
(Open 6 days 9am-5.30pm (Sat 10am-5.30pm)
ACORN/BBC DISTRIBUTOR
Forth and Pascal for BBC Micro and for Epson HX20. Printers, Disc Drives, Consumables.-

WARWICKSHIRE

LEAMINGTON HOBBY CENTRE
ACORN COMPUTER BBC
Warwickshire's sole official BBC Micro Dealer and Service Centre. Specialists in Monitors, Cumana and BBC Drives, and Epson Printers.
121 Regent Street, Leamington Spa.
TEL: (0926) 29211

DON'T LEAVE IT TO CHANCE. GIVE YOUR BUSINESS A BOOST BY ADVERTISING IN

A&B DEALERGUIDE

PHONE JASON ON 01-437 0699 for details.

A & B COMPUTING – CLASSIFIED ADVERTISEMENT – ORDER FORM

1.	2.	3.
4.	5.	6.
7.	8.	9.
10.	11.	12.
13.	14.	15.

Advertise nationally in these columns to over 100,000 readers for only 40p per word (minimum charge 15 words). Simply print your message in the coupon and send with your cheque or postal order made payable to Argus Specialist Publications Ltd to

CLASSIFIED DEPARTMENT A & B COMPUTING
No. 1 Golden Square, London W1.
01-437 0699

Name .....

Address .....

.....

.....

Tel. No. (Day) .....

Please place my advert in A & B Computing for ☐ issues. Please indicate number of insertions required.

To appear in the Nationwide Dealerguide at £27 per insertion, simply fill in the details below.

Company Name _____

Address _____

_____

Tel No & Contact _____

Additonal Copy _____

_____

_____

No of Insertions _____

Post to: A&B Computing, Classified Dept., ASP, 1 Golden Square, London W1 or phone 01-437 0699.